View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UC Research Repository

A Preliminary Study of a General Model for Supporting Tutorial Dialogues

Amali WEERASINGHE, Antonija MITROVIC and Brent MARTIN
Intelligent Computer Tutoring Group
University of Canterbury, Christchurch, New Zealand
{acw51, tanja, brent}@cosc.canterbury.ac.nz

Abstract: One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. We present a project with the goal of developing a general model for supporting tutorial dialogues that could be used in both well- and ill-defined instructional tasks. We have previously studied how human tutors provide additional support to students learning with an existing intelligent tutoring system. On the basis of these findings we developed a model for supporting tutorial dialogues, which we present in this paper. We used this model in a Wizard-of-Oz study to provide adaptive support. The results show that students did learn the relevant domain knowledge and that human tutors mostly agreed with the interventions generated from the model.

Keywords: tutorial dialogues, constraint-based tutors

1. Introduction

The main objective of the Intelligent Tutoring Systems (ITS) research community is to develop tutoring systems to achieve the effectiveness of one-on-one human tutoring, the most effective form of instruction [2]. One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. These dialogues provide opportunities for students to reflect on the existing knowledge and to construct new knowledge. Some of the dialogue-based tutoring systems that have been developed are Why2-Atlas [5], Auto Tutor [5], CIRCSIM-Tutor [7] and Geometry Explanation Tutor [1]. Why2-Atlas and AutoTutor use the dialogues as the main activity to help students learn the domain knowledge. The other systems provide problem-solving environments as the main activity and use tutorial dialogues as a way of remediating errors in the student solutions. For example, CIRCSIM-Tutor is a natural language (NL) tutor that helps students solve a set of problems in cardiovascular physiology relating to regulation of blood pressure. The Geometry Explanation Tutor requires students to justify the problem-solving steps in their own words. All these instructional tasks are well-defined: problem solving is well structured, and therefore explanations expected from learners can be clearly defined. In contrast, database design is an ill-defined task: the final result is defined only in abstract terms, and there is no algorithm to find it. In previous work we incorporated tutorial dialogues to our database design tutor [10].

Our long-term goal is to develop a general model for supporting dialogues across domains. Since we previously implemented dialogues for EER-Tutor [8], the initial work on this project started with the same system. As the first step, we conducted an observational study [9] focusing on how students interacted with EER-Tutor [8], while getting additional help from a human tutor through a chat interface. From the results of this study, we

developed a model to support dialogues, which we present in Section 2. In order to investigate the applicability of the model in a well-defined domain, another observational study was conducted, this time with the ERM-Tutor [6]. Section 3 describes the observational study and the results. Conclusions are given in the final section.

2. Prototype of the Model

Tutorial dialogues have been used in different pedagogical contexts based on the domain and the target student group. However, the problem-solving tasks supported were well-structured, and the types of explanations expected from students can be clearly defined. For example, in Mathematics and Physics, students are expected to explain the theorems that they have used. However, it is challenging to incorporate dialogues in an open-ended domain such as database design. It is not sufficient to ask the students to explain the concepts of database modeling, as the database design skills can only be developed through extensive practice. We also believe prompting them to explain every solution step will potentially place a heavy cognitive burden on the students. This may also demotivate natural explainers from using the dialogues. Hence tutorial dialogues are used to remediate errors in the student solution.

Our model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or more violated constraints, which form leaves of the hierarchy. The error types are then grouped into higher-level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue learners are able to provide answers by selecting the correct option from a list provided by the tutor. Each component is now described in detail.

2.1 Error Hierarchy

In previous work we developed a hierarchy of errors students make in the Entity-Relationship (ER) domain [10], which classifies all errors into syntactic and semantic in nature. A high level view of the hierarchy is given in Figure 1, showing the top three levels only. Syntax errors are generally simple, each requiring only one feedback message to be given to the student rather than initiating a dialogue; for that reason, every syntax error corresponds to a single violated constraint. For example, 58 constraints are associated with syntax errors for the ER domain; constraint 7 is violated when two entities are directly connected to each other. In contrast, the hierarchy for semantic errors is deeper because constraints are often related by some high-level concept. For semantic errors, error types are further divided into sub-errors (Figure 1) .

We were interested to investigate whether this hierarchy can be reused in other domains. For this investigation, we used the constraints from three constraint-based tutors: ERM-Tutor, which teaches mapping conceptual database schemas into relational ones [6], NORMIT which teaches data normalization and a fraction addition tutor. ERM-Tutor teaches logical database design which involves mapping an ER-schema to a relational schema using the 7-step mapping algorithm [4]. Data normalization is the process of refining a relational database schema in order to ensure that all relations are of high quality

[8]. All three domains are well-defined, because of the existence of algorithms to carry out each task.

ALL ERRORS
- Syntax errors
- Semantic errors
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Connecting an attribute to an incorrect construct
  - Errors dealing with cardinalities and participation

Fig. 1. Overall view of the error hierarchy

During this investigation, we identified situations when it was not enough to present a single feedback message for some violated syntax constraints: a dialogue was required. Therefore, we modified the structure of the error hierarchy to divide all error types into two main categories: *Basic Syntax Errors* and *Errors dealing with the main problem-solving activity* (Figure 2). Under the new node *Basic Syntax errors*, we included simple syntax errors, such as checking whether the student has filled the required fields, the components used to fill the required fields are valid etc. Hence it is sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted.

Another refinement required was to make the two domain-specific nodes *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* more general so that the overall hierarchy can be used across domains. As these two nodes deal with associations between solution components, it is appropriate to have a new node *Associations* (Figure 2). This new node has different domain-specific children. For the ER domain, *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* are child nodes of *Associations.*

The final refinement was made based on an observation from the previous study [9]: some students seem to be reacting to feedback on errors by making suggested changes without reflecting on other modifications that also need to be carried out. In ER modeling if a regular entity with a key attribute is changed to a weak entity a partial key should be specified instead of the key attribute. This may lead to frustration due to the number of attempts that the student has to go through to arrive at the correct solution. A new node *Failure to complete related changes* was added to the existing error hierarchy, which reminds the student to check whether other changes are necessary (Figure2). In such cases, the student will be prompted to reflect on other related changes before submitting the solution.

ALL ERRORS
- Basic syntax errors
- Errors dealing with the main problem solving activity
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Associations
  - Failure to complete related changes

Fig. 2. Overall view of the refined error hierarchy

Figure 2 only shows the top three levels of the error hierarchy; these levels are domain-independent and the lower levels deal with domain-specific concepts. The common feature in all these tasks is that the syntactic and semantic correctness of a solution can be completely evaluated by examining the components of the solution and their associations.

However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a paragraph, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used in a variety of instructional tasks where the solution can be completely evaluated by its components and their associations.

2.2 Tutorial Dialogues

In our model, error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e. each leaf node in the hierarchy), and discusses the domain concept associated with that error. Each dialogue consists of four stages. In the first stage, the dialogue informs the student about the concept that s/he is having difficulty with, and then asks for the justification of the student's action. The purpose of the second stage is to assist the student in understanding why the performed action is incorrect. The third stage prompts the student to specify how to correct the mistake. In the fourth stage, the student can review the domain concept learned.

EERTutor1: You seem to be having some difficulty with regular entities. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I don't know*
EERTutor2: Let's try a simple thing. How do we model characteristics of an entity?
Student2: *Using Attributes*
EERTutor3 : Great ! Now tell me why modelling *Grade* as an entity is incorrect?
Student3: *I'm not sure*
EERTutor4: Let's see. Check carefully whether you can find any attributes for *Grade*.
{After sometime} Pls. tell me whether you manage to find any attributes for *Grade*
Student4: *No*
EERTutor5: Fine. It doesn't have any attributes. So now can you tell me why modelling *Grade* as an entity is incorrect
Student5: *Because it doesn't have any attributes to describe its characteristics*
EERTutor6: Great! Now tell me the best way to model *Grade*?
Student6: *I need more help*
EERTutor7: Ok. If *Grade* doesn't have attributes, it cannot be an entity. So it should either be a relationship or an attribute. Now tell me the best way to model *Grade?*
Student7: *Attribute*
EERTutor8: Great ! Now use the correct word to fill in the blanks so that you'll remember what you learnt. …….. are used to describe characteristics of an …….. in a ER model
Student8: *attributes, entity*
EERTutor9: Excellent, go ahead and do the changes. Submit the solution when you're done.

Fig. 2: A hypothetical dialogue for the database design tutor

Figure 2 represents a hypothetical dialogue for the database design tutor. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). If the student fails to provide the correct answer (*Student1*), s/he will be asked a more specific question that provides a further opportunity to understand the domain concept that is violated (*EERTutor2*). However, if s/he fails to correct the mistake even after going through a series of detailed questions, as the last resort the tutor will provide an explanation on how to the correct the mistake together with a brief description about the domain concept that needs to be learnt (EERTutor4-8). The dialogues consist of simple questions (EERTutor1), fill-in-a-blank (EERTutor8), or true-false questions, to motivate the student to explain. When a certain mistake is repeated, the model informs the

student of its observations (EERTutor1), thereby providing an opportunity to reflect on his/her domain knowledge. As all dialogues facilitate the discussion of errors (EERTutor3), students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill. Although the prompts are domain-specific, the structure of the dialogues is domain-independent.

2.3 Rules for Adapting Dialogues

Adaptation rules enable individualization of the dialogues, by using the student model to decide on the timing, selection and entry point into the dialogue. Currently there are six rules and they are based on the observations from the study [9]. Some of the rules are discussed here. Rule 1 (dealing with timing of dialogues) checks whether the student made any attempts at the current problem, and has been inactive for a specified period of time (such as 1.5 minutes, the time period we observed in the study). If both these conditions are satisfied, then student's solution is evaluated even though it has not been submitted yet, and a dialogue is initiated to focus on the error most suitable for discussion if multiple errors exist.

Rule 3 addresses the critical issue of selecting a dialogue. Dialogue selection is very important because if it is not effective, it might be difficult for students to systematically develop a comprehensive mental model of the domain. Dialogue selection depends on the student solution and the error hierarchy. The probability of violating a constraint is calculated using the last five submissions on that constraint. For instance, if a constraint is violated twice in the last five submissions, the probability of not knowing it is 0.4. The probabilities of violating individual constraints are then combined to calculate the probability of making an error corresponding to higher-level nodes in the hierarchy. These probabilities are updated each time a student solution is evaluated. Rule 3 finds the error type (e.g. node N1, which is a non-leaf node in the hierarchy) that a student is most likely to make. As the nodes in the hierarchy are ordered from basic domain principles to more complicated ones, the dialogue associated with the left-most leaf node for N1 is chosen as the most suitable dialogue for a set of violated constraints.

Dialogues can be more effective if they are adapted to the student's domain knowledge and reasoning skills. We observed that the tutors tend to discuss the domain concepts relevant for an error if it was done repeatedly [9]. They also tend to state their observations before discussing the domain concept (e.g. "You seem to be having difficulty with regular entities (*EERTutor1* in Figure 2). Rule 4, which deals with the customizing the entry point to the dialogue, is activated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (*EERTutor1* in Figure 2). If the error was made less than *n* times, the dialogue will start from the error within the current context (*EERTutor3* in Figure 2). As these rules do not depend on the domain to individualise dialogues, the rules can be used across domains.

Even though this model was developed for constraint-based tutors, it can be used in any ITS providing a problem-solving environment. In such an ITS, a student solution is evaluated and feedback is provided on the errors regardless of the mechanism/methodology used for diagnosis. Therefore, the error hierarchy (the first component of the model) could be developed using the error types of that domain. Tutorial dialogues (the second component of the model) need to be written for each type of error based on the tutorial structure that was discussed in Section 2.2. The third component of the model, rules for adapting dialogues, are domain independent, hence it can be used across domains.

3. Preliminary Study

We conducted an experiment with the ERM-Tutor in April 2006 at the University of Canterbury, which involved volunteers from a database course and experienced tutors. Two types of feedback were provided: typical feedback provided by the system (i.e. hint sequences provided by ERM-Tutor), and dialogues initiated based on the model. The study was conducted as a Wizard-of-Oz study, in which the first author simulated the actions of the model. This additional assistance was given through a chat interface, and will be referred to as interventions hereinafter. Even though the first author used the dialogues from a written script, it was not always possible to use the scripted prompts in the later stages of dialogues. This is due to the student responses not being constrained as in the proposed system. (In the proposed system, the students will be given a list of possible answers from which the correct one can be selected).

Participants interacted with ERM-Tutor in one room, while the first author observed from another room. The participants could initiate interventions through the chat interface or the *More Help* button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, they filled out a questionnaire. The first phase of the study involved analyzing the logs to investigate the effectiveness of the dialogues. In the second phase, human tutors were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who indicated whether he/she agrees with the timing and the content of interventions. In the case of a disagreement, the judge was requested to provide justifications.

Ten students and five tutors (acting as judges) participated in the study. The judges were the lecturer and the tutors involved in teaching the course. The average session duration was 59 minutes (sd=15.3). Sometimes the ERM-tutor indicated that the student solution was incorrect even though it was actually correct, due to a coding problem. Such instances were excluded from the analysis. The average number of problems attempted was 11 (sd = 4.6), with 8.4 (sd = 5.2) completed.

From the logs, we identified 65 episodes, each pertaining to a single topic (as in [3]). In addition to facilitating remediation, some episodes focused on helping with the interface (such as moving to the next step), completing the session or helping with technical problems (e.g. web browser not being able to display the page). The number of episodes per session ranged from 1 to 13, with a mean of 6.5 (sd = 4.3). We are mainly interested in 31 episodes in which dialogues were facilitated. Six of these episodes contained a single utterance each, initiated by the wizard. For instance, a tutor utterance that helped a student to understand that multi-valued attributes are not mapped in the first step of the algorithm was "Think about the color attribute". The longest episode consisted of 11 utterances, 6 of which were provided by the model (i.e. the wizard). In the example dialogue (Figure 3), the student is incorrectly applying step 4 to the identifying relationship, while that step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

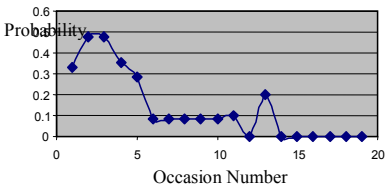

Fig.4. Probability of violating a constraint

In order to investigate whether the dialogues were effective, we analyzed how frequently an error occurred after being discussed in each episode. As the knowledge base in ERM-Tutor is represented as a set of constraints, the errors were recorded as violations of

ermtutor: what do you need to do when you're mapping a 1:N relationship?
coscstudent001: map the n-cardinality table
ermtutor: yes, what is the attribute that needs to be included
coscstudent001: the code from course
ermtutor: yes, good. but can you see this is a special case?
coscstudent001: because section is a weak entity?
ermtutor: yes

Fig. 3. A dialogue from the study

constraints. Thus we analyzed how frequently the constraints that were discussed in the dialogues were violated subsequently.

However, some students were able to correct errors themselves just before the episode started. In another situation, a student indicated that he did not require any assistance (even after a period of inactivity) when prompted. The remaining 15 (48.3%) episodes were included in this analysis. These dialogues involved only seven participants. (The dialogues with the other three students were among the ones excluded.) These dialogues were associated with seven different domain-level constraints. Figure 4 illustrates the learning curve for these constraints. The X-axis represents the occasion number (first, second and so on) when the student violated a constraint discussed in a dialogue subsequently. The Y-axis shows the probability of violating these constraints. The probabilities of violating a constraint on the first and subsequent occasions were averaged over all students. The curve is not smooth due to the small sample size (this analysis involved only seven constraints discussed in 15 dialogues with 7 participants).

However, the learning curve suggests that the probability of subsequently violating a constraint discussed in an episode decreases with occasion number. This indicates that the students seem to learn domain concepts discussed in the episodes, i.e. that the dialogues based on the proposed model did not have a detrimental effect on learning. In order to evaluate whether the remediation facilitated by this model actually enhances learning we need to compare the performance with a control group of students who interact with the system without the dialogues.

In phase 2, five judges analyzed the interventions and indicated whether they agreed with their timing and content. At the beginning of phase 2, they were informed that the goal of the study was to develop a model to facilitate remediation through tutorial dialogues while interacting with a tutoring system. The judges were asked to comment on the appropriateness of the timing and the content of interventions provided through the chat interface. The number of sessions analyzed ranged from 1 to 3 per judge. Due to time constraints, it was not possible to have every session investigated by two judges.

All the episodes were categorized by the rule that initiated them. Five rules were relevant in this study. Rule 1 (described in Section 2.3) was violated only once. Rule 2 (a variation of rule 1) which waits for 1 minute of inactivity after receiving feedback from the system at least once for the current step is violated three times. The tutors agreed with the timing and content of these interventions.

Rule 4 was relevant in 21 episodes and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. The judges disagreed with the content in three situations. For instance, a judge suggested using "*Is there a regular 1:N relationship to map in this problem?*" instead of the first prompt in Figure 3.

One of the issues to be addressed is how to facilitate remediation when nothing needs to be done in a particular step (Figure 3). According to our model, the initial prompt was "*What do you need to do when you're mapping a 1:N relationship?*" which may imply that the student needs to perform an action, even if there is nothing to do. It would be better if the prompt is changed to "*Do you know which type of relationship needs to be mapped in this step*?". The new prompt still discusses a domain concept so it still confines to the dialogue structure discussed in section 2.2.

Some judges preferred earlier interventions than those suggested by the model. The model waits for 3 repeated mistakes before initiating a dialogue. However, it might be effective to intervene after two repeated mistakes, because it is easier to assess what the student is trying to achieve in this particular domain. As the result, the number of times a mistake to be repeated before facilitating remediation may be domain-dependent. Rule 1, which checks for a period of inactivity may also be domain-dependant; however, there was no disagreement on these. Further investigation is needed before the model is changed.

4. Conclusions and Future Work

The research presented in this paper focuses on developing a model for supporting tutorial dialogues for error remediation for both ill- and well-defined tasks. A prototype model was developed based on the findings of a preliminary study using EER-Tutor, an ITS that teaches database design. Database design is an ill-defined task: the final result is defined only in abstract terms and there is no algorithm to find it. In order to investigate the reusability of the proposed model for well-defined tasks, we applied it in three other domains: ER-to-relational mapping, data normalization and fraction addition. We then refined the model based on the findings of this investigation.

This paper focuses on the study which used the model with ERM-Tutor, an ITS developed for the ER-to-relational mapping domain. In addition to the feedback provided by the system, error remediation was facilitated through a chat interface. The interventions through the chat interface were based on the model. Analysis of user logs indicates that students did learn the domain concepts discussed in the dialogues. Human tutors who were asked to analyze the dialogues mostly agreed with the interventions generated by the model. The findings from the reported study are being used to refine the model.

Our next step is to incorporate the model into both EER-Tutor and ERM-Tutor. The enhanced systems will later be evaluated in authentic classroom environments. The goal of these evaluations is to investigate whether the adaptive error remediation supported by the model is more effective in facilitating deep learning than the non-adaptive dialogues, in which two students (with different domain knowledge and reasoning skills) receive the same dialogue when they make the same types of errors in their solutions.

References

[1] Aleven, V., Ogan, A. Popescu, O. Torrey, Cristen, Koedinger, K. R. (2004) Evaluating the effectiveness of a Tutorial Dialogue System for Self-Explanation, Lester, J. C., Vicario, R.M., Papaguacu, F. (eds.)< Proc. Of ITS2004, pp 443-454

[2] Bloom, B. The 2-sigma problem: The search of group instruction as effective as one-to-one tutoring, Educational Researcher 13, 1984 pp.3-16

[3] Chi, M. T. H., S. A. Siler, et al. (2001) Learning from Human Tutoring. *Cognitive Science* 25, 471-533.

[4] Elmasri, R, Navathe, S. Fundamentals of Database Systems. Addison Wesley (2004)

[5] Grasser, A.C., VanLehn, K., Rose, C.P., Jordan P. W., Harter, D. Intelligent Tutoring Systems with Conversational Dialogue AI magazine, 39-51.

[6] Milik, N., Marshall, M., Mitrovic, A. (2006) Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds). Proc. 8th Int. Conf. on Intelligent Tutoring Systems, pp. 707-709.

[7] Millis, B., Evens, M. , Freedman, R. (2004), Implementing Directed Lines of Reasoning in an Intelligent Tutoring System using the Atlas Planning Environment, International Conference on Information Technology: ITCC 2004, pp. 729-733.

[8] Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. (2004) DB-suite: Experiences with Three Intelligent Web-based database Tutors . Interactive Learning Research, 15(4), 409-432

[9] Weerasinghe, A., Mitrovic, A. (2006) Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based tutors, Aleven V. Ashley, K., Lynch, C. and Pinkwart, N. (eds.), Workshop on ITS for ill-defined domains at ITS2006, pp. 55-64

[10] Weerasinghe, A., Mitrovic, A. (2006) Facilitating Deep Learning through Self-Explanation in an Open-ended Domain, Knowledge-based and Intelligent Tutoring Systems 10(1), 3-19